Group 11 CS4246 Mini-Project Writeup

Team Member 1: Solayappan Meenakshi A0172687N
Team Member 2: Liu Zechu A0188295L
Collaborators: None
Sources: None

Approach one:

We used Deep Q-Learning (DQN) as the primary approach to solve the task. The neural network used was the AtariDQN network provided in `models.py`.To solve the problem of sparse reward and to speed up learning, we used a combination of reward shaping and curriculum learning.

Reward shaping:

In addition to the original reward signals, we added our own reward signals to guide the agent in the right direction to explore initially. We defined two quantities:

(1) `manhattan_distance`: the x-distance from the agent to the goal plus the y-distance from the agent to the goal
(2) `number_of_occupied_neighbours`: the number of neighbouring cells which are currently occupied by a car or its occupancy trail. Only up, front and back neighbours are considered. Thus, the range of `number_of_occupied_neighbours` is 0 to 3 (inclusive).

Our modified reward signal is as follows:

```
reward = original_reward +
      [(-1 * mahattan_distance) +
       (-5 * number_of_occupied_neighbours)] * 0.05
```

Intuitively, our reward shaping encourages the agent to go towards the general direction of the goal while trying to stay away from other cars.

Curriculum learning:

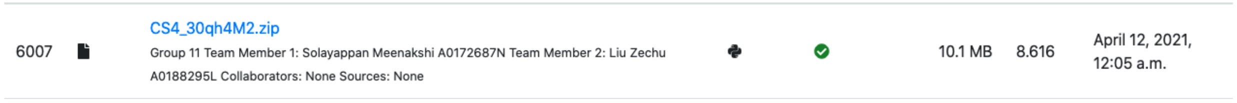
We designed a curriculum and iteratively trained the agent on increasingly challenging environments. The model saved after each training was used to initialise the model for the subsequent training session. Note that in each of the following training sessions, the agent achieved steadily increasing rewards towards the end. Also, the epsilon value decayed from 1.0 to 0.1 in each session.

(1) First training: 2000 episodes **with reward shaping**. Only **1 car** in each lane.
(2) Second training: 2000 episodes **with reward shaping**. Only **2 cars** in each lane.
(3) Third training: 2000 episodes **without reward shaping**. Only **4 cars** in each lane.
(4) Fourth training: 2000 episodes **without reward shaping**. Only **6 cars** in each lane.
(5) Fifth training: 6000 episodes **without reward shaping**. Trained on the full environment specified in `env.py`. Also, we randomised environment generation at the start of each episode.

After 5 rounds of training, the agent scored 7.916 on the aiVLE server.

(6) Sixth training: 10,000 episodes **without reward shaping**. Trained on the full environment specified in env.py. Randomised environment generation. Epsilon was fixed at 0.01 throughout all the episodes.

After the final round of training, the agent scored 8.616 on the aiVLE server (>85% success).



6007 CS4_30qh4M2.zip
Group 11 Team Member 1: Solayappan Meenakshi A0172687N Team Member 2: Liu Zechu A0188295L Collaborators: None Sources: None
10.1 MB 8.616 April 12, 2021, 12:05 a.m.

Approach two:

In addition, we also implemented Imitation Learning from expert actions generated by PDDL solver. To aid initial learning using DQN, the agent copies PDDL solver action with epsilon probability. For PDDL solver, since the car speeds are stochastic, we assumed the worst case scenario and chose the upper bound of that lane's speed to solve the problem.

Approach 2A:

For PDDL solver, a new plan would be generated at each step/after each action is taken based on the new state that is given. Use the final goal as the goal state to generate a plan. If no plan is possible, take action suggested by DQN model.

There was a cap on the maximum number of expert PDDL actions that could be taken. When the maximum number of PDDL actions were capped at 400, the agent scored 3.768 and did slightly better with a score of 4.284 when they were capped at 800.

Approach 2B:

Improvement on approach 2A which took a very long time to train and often has no possible plan which can be generated. Since the state is changing with every step, it is unnecessary to plan all the way to the final goal state. Hence, if the agent is sufficiently far away from the goal, generate a suitable subgoal which is closer to the agent and in the general direction the final goal. There are two things to note:

1. Since subgoal is closer to the agent than the final goal, the problem will be solved much faster
2. Since the subgoal is generated after each action/step a single generated subgoal will not affect the overall path that the agent takes that much. Moreover, the subgoal generated is sufficiently far away from the agent such that the agent cannot reach the subgoal in one action (i.e. agent does not need to pass through subgoal to reach final goal). This is to prevent penalising the agent should the subgoal chosen be a bad choice.

For Approach 2B, when the maximum number of PDDL actions were capped at 400, the agent scored 3.639 and did slightly better with a score of 4.485 when they were capped at 800. Approach 2B does not seem to fare much better than approach 2A apart from the decrease in training time. Since approach 1 significantly outperforms approach 2, we submitted approach 1.